Investigating Fine-Tuning for Irregularly Sampled Time Series

Yu Hei Chau
20644747
HKUST
louis321yh@gmail.com

Abstract

Forecasting irregularly sampled time series is of important in various fields, such as medical data analysis. Typically, a large batch of samples can be found within a domain, allowing for the pretraining of general models and then finetuning specialized models tailored to new datasets with some domain shift. We examine the possibility of an irregularly sampled time series forecasting model to learn from general data and subsequently fine-tune itself on a new dataset through domain adaptation.

1 Introduction

Forecasting irregularly sampled time series plays a crucial role in a wide range of applications, particularly in the realm of medical data analysis. In such scenarios, data points are often collected at irregular intervals due to the inherent unpredictability of events, such as patients' visits and emergency room admissions, or the physiological indicators taken by patients with some healthcare devices. Accurate and reliable predictions in this context can greatly aid healthcare professionals and patients in making informed decisions and improving patient outcomes.

To tackle the challenges of forecasting for irregularly sampled time series, researchers have proposed using Neural Ordinary Differential Equation (NODE) architectures [1] [2]. These innovative models are designed to address the irregular time steps associated with irregularly sampled data, by smoothly interpolating between the time steps using learned ODE dynamics, offering a promising solution for accurate forecasting in various problems with irregularly sampled time series. However, Neural ODEs come with their own set of limitations, primarily the long training time required due to the incorporation of a numerical ODE solver in the architecture, which requires a lot of number of function evaluations during training and inference [3].

In many cases, a large batch of samples is available within a specific domain, which serves as a valuable resource for training forecasting models. Utilizing this wealth of data, researchers can develop specialized models that account for the unique characteristics of the domain. However, when faced with a new set of data that exhibits some degree of domain shift, these models must be adapted to accurately capture the underlying patterns and dynamics in the new set of data.

Given the time-consuming nature of training Neural ODEs, it is not practical to train a new model for each new domain, particularly when the new domain shares similarities with the domain of the large batch of samples. In such cases, domain adaptation techniques can be employed to fine-tune the pre-trained model on the new dataset. This approach raises the question of whether a Neural ODE model can learn from general data and subsequently refine its predictions on a new set of data with distinct characteristics, while still maintaining reasonable training times.

By exploring the potential of Neural ODEs in irregularly sampled time series forecasting and leveraging domain adaptation, researchers could develop more versatile and efficient models. By

Not on the 37th Conference on Neural Information Processing Systems (NeurIPS 2023).

enabling forecasting models to adjust to similar domains with minimal retraining, we can overcome the constraints of lengthy training times, rendering them more feasible for real-world applications, particularly those with restricted computational resources.

In the upcoming sections of this report, we delve deeper into the topic of fine-tuning for irregularly sampled time series. Section 2 provides a comprehensive review of relevant literature, covering both fine-tuning techniques and the challenges of handling irregularly sampled time series. In Section 3, we introduce a toy dataset, which serves as an illustrative example to elucidate the problem of fine-tuning in the context of irregularly sampled time series. The experimental setup and results are presented in Section 4, where we assess the performance of various techniques on our dataset. Finally, in Section 5, we synthesize our findings, draw conclusions from the experimental results, and discuss their implications, while also proposing further hypotheses to explore in future research.

2 Relevant work

2.1 Irregularly sampled time series

A variety of traditional data-driven methodologies have been devised for the analysis of sequential data, including Recurrent Neural Networks and their various offshoots [4] [5] [6] [7], which have demonstrated success in detecting temporal dependencies in time series tasks. Nonetheless, in the presence of irregularly sampled time series data, additional challenges emerge.

To surmount these obstacles, researchers have proposed several inventive techniques. Some methods incorporates the time step difference as an input for the RNN, while others try to impute data in the missing time-steps [8]. Another approach entails utilizing exponential decay of the hidden state between observations, enabling the model to adjust its memory retention based on the time elapsed since the previous observation [9].

A newer category of models called Neural Ordinary Differential Equations (Neural ODEs) has surfaced [1], offering a promising solution for handling continuous time series data. Neural ODEs model the time series with an ODE, employing a Neural Network to learn the vector field for the ODE dynamics [1]. This methodology has resulted in the creation of various Neural ODE variants [2] [10], specifically engineered to address the analysis of irregularly sampled time series data.

One such variant, Latent ODEs [2], extends the exponential decay hidden state transition dynamics by replacing the exponential decay with learnable Neural ODE dynamics. This modification allows the model to capture intricate temporal patterns in irregularly sampled data more effectively.

2.2 Fine-tuning of pretrained models

Latent ODE models, or Latent Ordinary Differential Equation models, can be interpreted as sequence-to-sequence models for continuous or irregularly sampled time series data [2]. Meanwhile, sequence-to-sequence models are also commonly employed in NLP, such as Encoder-Decoder models using Recurrent Neural Networks (RNNs) or Transformers, focus on translating a sequence of text into another language [11] [12].

Fine-tuning and transfer learning are techniques commonly employed in NLP to adapt pre-trained models to specific tasks, such as those of sentence embedding problems [13] [14]. Given the similarities between Latent ODE sequence-to-sequence models and NLP sequence-to-sequence models, it is natural to wonder whether fine-tuning is feasible for Latent ODE models.

3 Example: Glucose Time series

3.1 Motivation

Blood glucose forecasting enables patients to track their glucose level and make informed decisions about their treatment plans. Patients can measure their current blood glucose level using a device, but these measurements are often taken irregularly due to various factors, such as the patient's schedule or the availability of the device. As a result, the glucose data collected constitute an irregularly sampled time series, which presents challenges for forecasting.

Table 1: Parameters for individual time series

Symbol	Usage
x_i	The onset of each glucose spike
w_i	The height of each glucose spike
d	The decay rate of each glucose spike

A large batch of glucose level data from many patients could be obtained through clinical trials, providing a valuable resource for training forecasting models. There are numerous similarities across the domain of glucose time series, such as the periodicity of glucose levels due to physiological factors like the circadian rhythm [15], and the onset of glucose spikes following a meal, which occurs at approximately random onsets [16]. However, when considering the glucose time series of a new patient not included in the previous batch, it is expected that there is some domain shift due to factors such as individual glucose tolerance, average glucose intake per meal, and other underlying physiological factors.

Neural ODE models have shown promise in addressing the challenges of irregularly sampled time series forecasting [2] [10], but their long training times present a problem for end devices with limited computational capabilities. It is important to note that most handheld consumer devices are not equipped with powerful GPUs, which are typically required for training complex models like Neural ODEs. This hardware limitation further emphasizes the need for an alternative approach to accommodate the constraints of these devices.

Additionally, users may prefer to keep their glucose data locally on their device for privacy reasons, rather than uploading it to a cloud server. In this context, pretraining a Neural ODE model using the large batch of data from clinical trials and then fine-tuning it with the user's data could provide a solution to the issue of slow training on the device.

By leveraging domain adaptation techniques and fine-tuning the pretrained Neural ODE model with the user's glucose data, the model can be adapted to the user by accounting for the domain shift while maintaining reasonable training times. This approach would allow for the development of more personalized and efficient forecasting models for blood glucose levels, which can be deployed on individual devices without the need for powerful GPUs or compromising user privacy. As a result, patients could benefit from accurate and timely predictions, leading to improved diabetes management and overall health outcomes.

3.2 Toy dataset time series formula

We use a toy dataset which tries to resemble the properties of glucose time series. These time series were created to include spikes with random onset times and varying heights, both corresponding to glucose intake due to meals. We also incorporate a decay rate parameter that influenced the decay rate of the glucose spikes, which roughly corresponds to glucose tolerance.

For each generated time series, a random time point was selected, and the series was split into two distinct parts. The first part, comprising the time series preceding the chosen time point, was used for regression, while the second part, consisting of the remaining time series, was designated for forecasting. To simulate real-world conditions, the time series before the time point was sampled irregularly.

The time series are generated with the parameters in Table 1 using the formula

$$f_{\textit{params}}(t) = \sum_{i=1}^{4} 1.2 w_i \exp(-d^2(\frac{t - x_i}{0.4})^2)\sigma(8 \cdot \frac{t - x_i}{0.4})$$

where σ is the sigmoid function. Note that there are four spikes for each time series, and the decay rate is independent of the spikes in the same time series. We describe the rules for sampling the parameters in the datasets below.

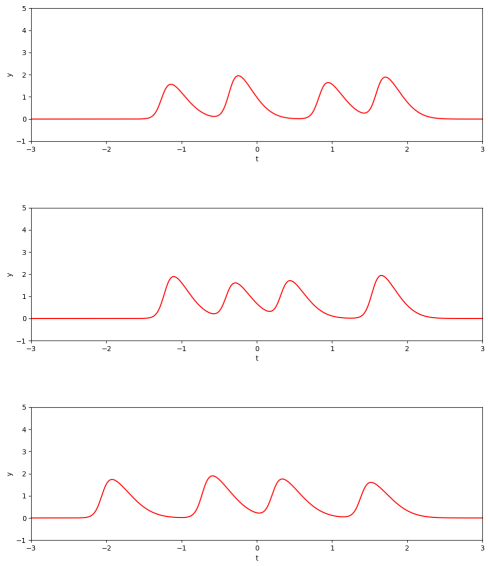


Figure 1: Samples from batch dataset

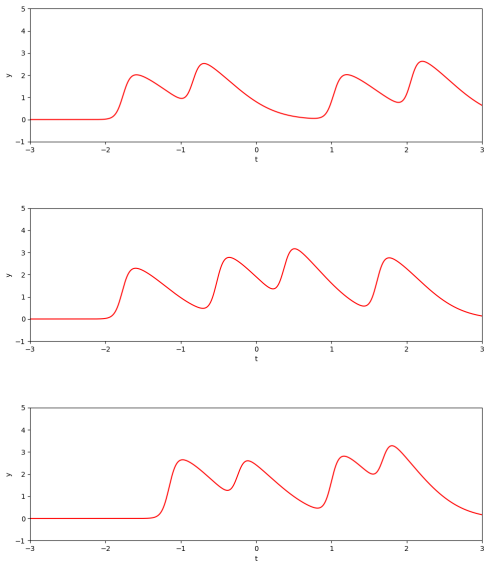


Figure 2: Samples from finetuning dataset

3.3 Toy dataset generation

More examples of samples can be viewed via the code glucose_series_samples_display.py, with cmd args –sample_from_finetuning_set to choose the dataset to sample from. The onset x_i are sampled:

$$y_1, y_2, y_3, y_4 \sim \text{Unif}([0, 1]) \tag{1}$$

$$x_1 = -y_1 - y_2 - 0.6 \tag{2}$$

$$x_2 = -y_2 - 0.2 \tag{3}$$

$$x_3 = y_3 + 0.2 \tag{4}$$

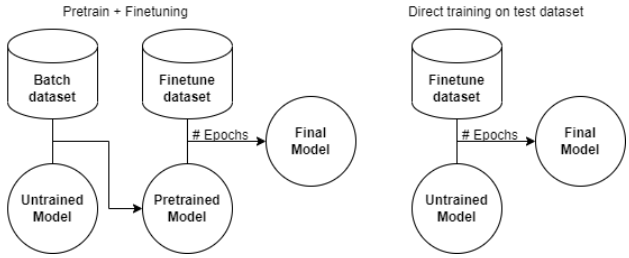
$$x_4 = y_3 + y_4 + 0.8 \tag{5}$$

Batch dataset The batch dataset will be represented by "four people". We randomly pick a set $D \subset [0.7, 0.11]$ with $|D| = 4$, and the decay values for each time series will be picked randomly $d \in D$. Each weight $w_i \sim \text{Unif}([1.5, 2])$.

Finetuning dataset The finetuning dataset will be represented by "a single person". Here $d = 0.5$ always, and each weight $w_i \sim \text{Unif}([1.8, 2.5])$. This roughly corresponds to a domain shift, where the new person has "more insensitivity to glucose" so the decay rate is less, and person is likely to eat more food per meal hence the on average larger weights. These differences are shown qualitatively in Figures 1 and 2.

4 Experiments

We conduct an experiment to investigate the potential of finetuning in the context of glucose time series, using the Latent Neural ODE model by [2], and the aforementioned toy dataset. We compare the two training methods:



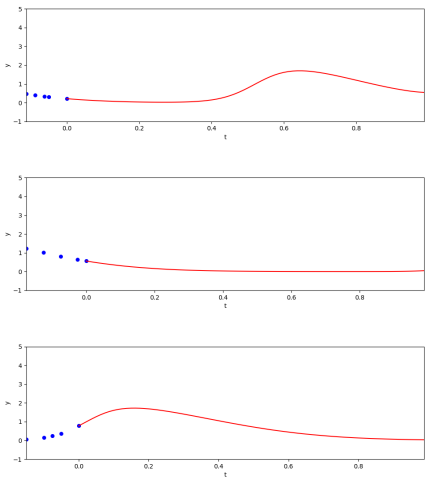


Figure 3: Examples of batch dataset

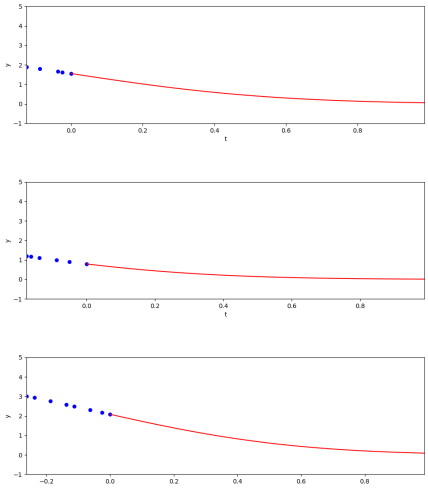


Figure 4: Samples from finetuning dataset

We keep the model architecture and the loss function the same in both training processes. The pretraining step is trained for 5000 epochs to make sure the model converges on the batch dataset. For the finetuning step in Pretrain + Finetuning, a learning rate schedule with 50 linear warmup steps from $\alpha = 0$ to $\alpha = 0.01$ is employed, and the learning rate decays exponentially until $\alpha = 0.0001$. The learned weights from the model obtain via the pretraining step are used as the initial weights for the finetuning step. The model architecture is kept the same.

For the direct training method, a fixed learning rate of $\alpha = 0.02, 0.01, 0.005, 0.003, 0.001$ are used (we train 5 models), to control for the slower convergence due to low learning rate ($\alpha = 0.02$ is larger than the maximum learning rate in the schedule for Pretrain + Finetuning).

We record each model in each epoch starting from the training of the model with the finetuning dataset. For Pretrain + Finetuning, the 5000 epochs for the pretraining step are not counted, as we are concerned with whether pretraining will accelerate a model's training performance on downstream tasks.

4.1 Model specifications

The model is essentially the model as described in [2]. We use the torchdiffeq package by [1] for the Neural ODE dynamics. Our implementation of the LatentODE model consists of an encoder,

$$\text{Vector embedding} = \text{Encoder}(\{(t_i, y_i)\}_{i=1}^{N})$$

where N is not necessarily the same for different inputs for the same model, and outputs a prediction

$$\{(y_i')\}_1^M = \text{Decoder}(\text{Vector embedding})$$

on the time steps $\{t_i'\}_1^M$ such that $t_0' = \max_i t_i$ and $t_0', \cdots, t_M'$ are uniformly spaced time points. The details of the LatentODE model are left to the original paper in [2].

4.2 Training

We expect the models to not do so well on predicting the onset of the next glucose spike, since it is fundamentally random. Therefore we try to do a "short time" prediction, to see whether the model is successful in predicting the glucose levels within the context of a single spike.

We choose a random interval within $[-3, 3]$ and split it up, the points before the prediction start time t_N are irregularly sampled (denoted blue), and the objective is to predict the trajectory after t_N (denoted red) in Figures 3 and 4. We use a weighted L1 loss between the prediction and objective, favoring predictions in the short term.

4.3 Results

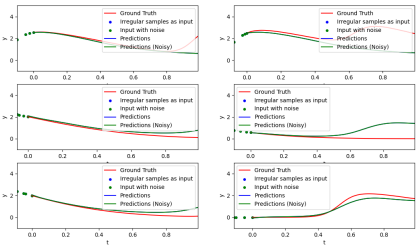
Table of Losses		Epochs=				
		100	200	300	400	500
Training Method	Finetuning	0.143014	0.138858	0.13749	0.136129	0.135166
	Direct $\alpha = 2 \cdot 10^{-3}$	0.206081	0.20193	0.183519	0.193032	0.169423
	Direct $\alpha = 1 \cdot 10^{-3}$	0.209279	0.193541	0.203403	0.188115	0.180546
	Direct $\alpha = 5 \cdot 10^{-4}$	0.215053	0.196338	0.188313	0.184971	0.182462
	Direct $\alpha = 3 \cdot 10^{-4}$	0.232216	0.203526	0.193772	0.188427	0.184207
	Direct $\alpha = 1 \cdot 10^{-4}$	0.312492	0.238092	0.217222	0.20625	0.201295

Table 2: Table of losses with respect to finetuning dataset

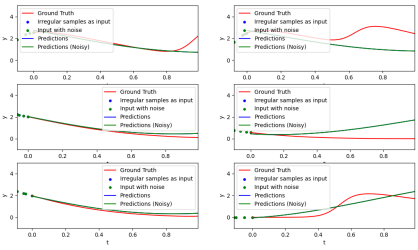
In table 2, we sample 200000 instances of toy time series from the finetuning dataset, and compute the mean L1 prediction loss over the 200000 samples.

$$\text{Mean loss} = \frac{1}{200000} \sum_{\text{sample}} \text{Weighted } L^1 \text{ Loss}(\text{Actual}_{\text{sample}}, \text{Predicted}_{\text{sample}})$$

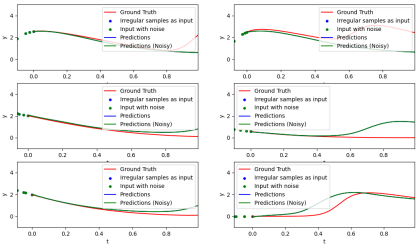
where $\text{Actual}_{\text{sample}}$ and $\text{Predicted}_{\text{sample}}$ are the corresponding time series for a sample specified by Section 3.3. We depict some examples of prediction sampled from the batch dataset for the Pretraining + Finetuning model and the directly trained model with $\alpha = 1 \cdot 10^{-3}$ (zoom in the PDF to read the graphs in detail).



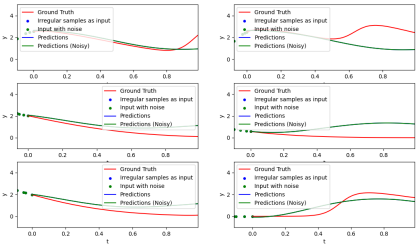
Finetuning Epoch=100



Direct Epoch=100



Finetuning Epoch=300



Direct Epoch=300

In the above graphs, the model is given the green points as input, and the green line is the model's prediction. The blue data are not shown since the noise for the input points are set to 0, so both data overlap. The program for visualizing these graphs is described in the Appendix.

5 Conclusions and Limitations

Conclusions Table 2 shows that the approach from finetuning indeed achieves a lower loss than training directly from the new data, regardless of the learning rate α used. From inspecting the images, it seems that the finetuning approach can remember what happens to a glucose spike when it about to reach a peak, but the models trained from scratch has to learn that from the beginning.

This behavior is shown in the four examples above, the finetuning model is able to approximate the curve where the information of the previous time points indicates a rising glucose level (first row of the six images), where the model estimates the height of the spike and lets it decay. However, the directly trained model at epoch=100 approximates the behavior by interpolating the spike with a straight line (the predicted time series does not first go up and then curves downwards). At epoch=300 it is able to approximate this behavior.

At epoch=300 the directly trained model seems to do worse than epoch=100 on the two bottom left images, which might be due to the fact that the model has compensated for minimizing other samples while sacrificing these two samples. This is however not the case for the finetuning approach. This may suggest that the finetuning approach is able to achieve faster convergence.

This shows that it may be possible to leverage fine-tuning and domain adaptation, which are common techniques in NLP, to the problem of predicting irregularly sampled time series.

Limitations The results are not extensive enough. Some more analyses can be done on this toy dataset, such as training the 6 models until they converge and measure the epochs needed for convergence and their losses. Also, the model is currently trained for directly predicting the time series without any noise for the input, which does not allow us to evaluate whether the Latent ODE model is robust to noise, and whether this property could be enhanced via pretraining + finetuning. This allows us to have a better sense on how much pretraining + fine-tuning outperforms training directly, on this toy dataset.

Moreover, the training and the evaluation metrics are "essentially" computed with the same dataset, since they are sampled using the same method. In this sense, Table 2 is only a training loss metric, but not some evaluation loss on some hidden data.

It could be argued that time series prediction tasks implicitly assume that $y(t)$ are sampled from the same distribution for the same entity, such as the glucose time series depends on the hidden dynamics of a person, which has hidden parameters that come from the same distribution describing the person. It could also be argued that having a lower training loss indeed shows the model does converge faster to a minimum.

Nonetheless, it would be much better if there is access to an actual batch of glucose time series to work with, so that proper train/test split could be done (such as training on the new person only times $t < T$, and use times $t > T$ for inference), and to verify the idea of training+finetuning on an actual dataset.

6 Acknowledgements

I thank Prof. Gary Chan (HKUST CSE) for providing me an opportunity to explore data analysis topics via COMP4971. I also thank his PhD student Shuhan Zhong (HKUST) for various discussions about recent developments in the time series prediction problem, and the help he has given to proofread this report and his comments and suggestions on improving the academic writing style.

References

[1] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," in *Advances in Neural Information Processing Systems*, 2018.

[2] Y. Rubanova, R. T. Q. Chen, and D. K. Duvenaud, "Latent ordinary differential equations for irregularly sampled time series," in *Advances in Neural Information Processing Systems 32*, 2019.

[3] H. Xia, V. Suliafu, H. Ji, T. M. Nguyen, A. Bertozzi, S. Osher, and B. Wang, "Heavy ball neural ordinary differential equations," in *Advances in Neural Information Processing Systems*, 2021.

[4] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," *arXiv preprint arXiv:1801.01078*, 2017.

[5] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[8] D. M. Kreindler and C. J. Lumsden, "The effects of the irregular sample and missing data in time series analysis." *Nonlinear dynamics, psychology, and life sciences*, 2006.

[9] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific reports*, vol. 8, no. 1, p. 6085, 2018.

[10] P. Kidger, J. Morrill, J. Foster, and T. Lyons, "Neural controlled differential equations for irregular time series," in *34th Conference on Neural Information Processing Systems*, 2020.

[11] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[14] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[15] E. Van Cauter, K. L. Knutson, R. Leproult, and K. Spiegel, "Roles of circadian rhythmicity and sleep in human glucose regulation," *Endocrine Reviews*, vol. 18, no. 5, pp. 716–738, 1997.

[16] G. Freckmann, S. Hagenlocher, A. Baumstark, N. Jendrike, R. C. Gillen, K. Rössner, and C. Haug, "Continuous glucose profiles in healthy subjects under everyday life conditions and after different meals," *J Diabetes Sci Technol*, vol. 1, no. 5, pp. 695–703, 2007.

7 Appendix - Description of code

7.1 Model training

The models are trained via glucose_series_testing.py and glucose_series_finetuning.py. We do not train a variational Encoder Decoder, no random sampling in the latent space is done. The models are therefore trained without the –variational and –reduce-dims args.

Pretraining / training The glucose_series_testing.py contains code to train models from scratch. The –use_alternative_dataset flag indicates whether the Batch dataset or Finetuning dataset (see Section 3.3) is used. If it is present, we use the Finetuning dataset.

Finetuning The glucose_series_finetuning.py contains code to load pretrained weights from a model and then finetune the model on the Finetuning dataset.

7.2 Visualizing examples of sampled time series

The glucose_series_samples_display.py contains code to display interactively random samples drawn from the Batch dataset distribution and Finetuning dataset distribution. The argument –sample_from_finetuning_set indicates which distribution to sample from.

7.3 Visualizing model prediction

The glucose_series_inference.py contains a Gradio app for visualizing the time series prediction of trained models (such as the Figures in Section 4.3). The models could be loaded in the Choose model tab. The images and settings are in the inference tab.

The random noise slider dictates the standard deviation for the random Gaussian noise for the input samples $\{(t_i, y_i)\}_1^N$. The noise z_i is added to y_i. The number of samples slider adjusts the value N.

The relative separation slider adjusts the coefficient for on average how close the time points $\{t_i\}_1^N$ for the samples should be. The figures in Section 4.3 are obtained with relative separation $= 1$, meaning that it is the same as the training. The decay coefficient slider fixes the value d in Section 3.

The large glucose intake checkbox indicates whether $w_i \sim \text{Unif}([1.5, 2])$ or $w_i \sim \text{Unif}([1.8, 2.5])$. The figures in 4.3 are drawn with this enabled. For more details of w_i see Section 3.

For the buttons, the regenerate random noise button randomly readds noise to the fixed sample points. The regenerate random samples button resamples both the points $\{(t_i, y_i)\}_1^N$ and the noise corrupted points. The predict with new model button recomputes the predictions, with the input points fixed, whenever another model is loaded.

7.4 Computation of loss metrics

The loss metrics are computed via glucose_series_metrics.py.